Videojuegos

Curso de Diseño y Programación

Nº 20 5,99 euros



Completamos el **patrón rítmico**

Instalando el juego con **Inno Setup**

MULTIPLAYER SETUP

PLAYER SETUP

20







Curso práctico de Videojuegos

AUTOR DE LA OBRA
Marcos Medina
DIRECCIÓN EDITORIAL
Eduardo Toribio
etoribio@iberprensa.com
COORDINACIÓN EDITORIAL
Eva-Margarita García
eva@iberprensa.com
DISEÑO Y MAQUETACIÓN
Antonio Gª Tomé
PRODUCCIÓN
Marisa Cogorro
SUSCRIPCIONES
Tel: 91 628 02 03
Fax: 91 628 09 35
suscripciones@iberprensa.com
FILMACIÓN: Fotpreim Duvial
IMPRESIÓN: Gráficas Don Bosco
DUPLICACIÓN CD.ROM: M.P.O.
DISTRIBUCIÓN
S.G.E.L.
Avda. Valdelaparra 29 (Pol. Ind.)
28108 Alcobendas (Madrid)
Tel.: 91 657 69 00

EDITA: Iberprensa
www.iberprensa.com



CONSEJERO
Carlos Peropadre

REDACCIÓN, PUBLICIDAD Y ADMINISTRACIÓN
C/ del Río Ter, 7 (Pol. Ind. "El Nogal")
28110 Algete (Madrid)
Tel.: 91 628 02 03
Fax: 91 628 09 35
(Añada 34 si llama desde fuera de España.)

DEPÓSITO LEGAL: M-35934-2002
ISBN: Coleccionable: 84 932417 2 5
Tomo 2: 84 932417 4 1
Obra Completa: 84 932417 5 X

NOTA IMPORTANTE:
Algunos programas incluidos en los CD de "Programación y Diseño de Videojuegos" son versiones completas, pero en otros casos se trata de versiones demo o trial, versiones de evaluación que Iberprensa quiere ofrecer a nuestros lectores. No se trata en ningún caso de las versiones comerciales de los programas, y las hemos incluido para dar al lector la oportunidad de conocer y probar esos programas y que así pueda decidir posteriormente si desea o no adquirir las versiones comerciales de cada uno.

Fin de la obra

Terminamos con este número nuestra obra dedicada al diseño y la programación de videojuegos, dedicada a formar al alumno en las principales técnicas relacionadas en el desarrollo completo de un videojuego 3D.

A lo largo de la obra hemos aprendido programación a nivel general y a nivel específico con ciertas herramientas y lenguajes. También hemos trabajado con distintas aplicaciones de edición y retoque de imágenes, tanto en 2D como en 3D. Hemos descubierto las aplicaciones más importantes para crear nuestra propia música y toda clase de efectos sonoros, y hemos aprendido la historia de los videojuegos desde sus inicios hasta el panorama actual.

Además, con esta obra se ha pretendido ayudar a fomentar la creatividad del lector, dándole ideas para sus desarrollos y diseños y animándole siempre a practicar, pues así es como se consiguen los mejores resultados.

Terminamos aquí el viaje de 20 semanas articulado en 400 páginas y 20 CD-ROMs, en el que deseamos que hayas adquirido una buena base para poder continuar a partir de ahí.

Aprovechamos para agradecer tu confianza, esperamos que **Programación y Diseño de Videojuegos** haya sido una obra provechosa y que, en definitiva, te hayas divertido.

Porque en el fondo, de eso se trata.

sumario



PARA ENCUADERNAR LA OBRA:

- ***Tapas del volumen 1 disponibles en Iberprensa. Pedidos por teléfono: 916280203.***
- ***Tapas del volumen 2 ya a la venta en quioscos.***
- ***Los suscriptores recibirán las tapas en su domicilio sin cargo alguno como obsequio de Iberprensa.***

SERVICIO TÉCNICO:
Para consultas, dudas técnicas y reclamaciones Iberprensa ofrece la siguiente dirección de correo electrónico: games@iberprensa.com

PETICIÓN DE NÚMEROS ATRASADOS:
El envío de números sueltos o atrasados se realizará contra reembolso del precio de venta al público más el coste de los gastos de envío. Pueden ser solicitados en el teléfono de atención al cliente 91 628 02 03

Sistema multijugador

Concluimos el desarrollo de nuestro juego "Zone of Fighters" implementando un sencillo módulo multijugador usando el protocolo UDP para dos ordenadores conectados entre sí por red local.

Como sabemos, este protocolo es ideal para intercambiar datos entre ordenadores a gran velocidad, necesario para juegos en tiempo real como "Zone of Fighters". Hemos implementado el módulo multijugador en un programa totalmente independiente, para facilitar su comprensión. La idea básica es que uno de los ordenadores funcione como anfitrión del juego (Host) y el otro como invitado o remoto. La jugabilidad o gameplay se basará en varios modos de juego en el que las dos bionaves lucharán en un mismo terreno contra un número determinado de ovnis enemigos y según el juego elegido, entre ellas también.

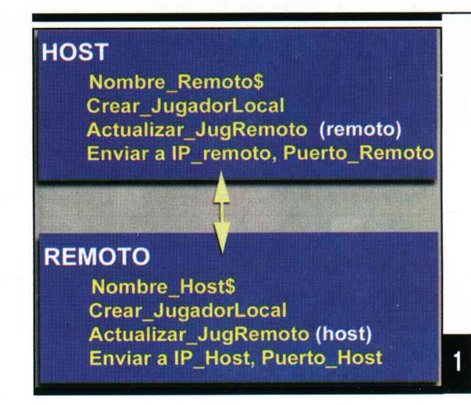
(■) CONCEPTOS BÁSICOS DE DESARROLLO

Antes de programar cualquier juego con algún sistema multijugador hay que tener muy claro qué es lo que se quiere, ya que testear este tipo de programas depende de una instalación adecuada. En primer lugar, tenemos dos ordenadores que envían y reciben paquetes de información entre sí a través de un mismo programa. Por lo tanto, este programa debe hacer distinción entre quién es el host y quién el remoto. Para ambos casos, debemos realizar un código diferente pero manteniendo una estrecha relación entre sí. Aparte, el resto del código será de uso común como: recursos, la relación con el entorno, la creación de disparos, sistema de partículas, visualización de indicadores, etc. Los paquetes de información contienen datos acerca de la posición y estado de ambas bionaves y su flujo debe mantener un cierto orden para evitar cortes de conexión. Además, debemos controlar al máximo la cantidad de bytes de cada paquete enviado para aumentar el rendimiento de respuesta. Por eso, solo se incluirá información fundamental de cada jugador imposible de obtener. El resto de información se debe procurar obtener en base a los datos disponibles. Todo paquete debe ir encabezado con un byte que indique de qué tipo es, ya que en el protocolo UDP no va identificada. Por último, el host es el encargado de crear el juego y sus características. Por lo tanto, el jugador remoto se adapta a ellos como invitado a la partida.

(■) MÓDULO PRINCIPAL. PREPARANDO EL JUEGO

Como novedad al código de "Zone of Figthers" para un jugador, disponemos de varios módulos diferentes que explicamos en esta entrega: módulo principal ("Multijugador.bb"), módulo de control del jugador local (host o remoto) ("JugadorLocal.bb"), módulo de control del juego ("Juego.bb"), módulo de control de ovnis enemigos ("Ovnis.bb") y módulo de control de red ("Red.bb").

En el módulo principal ("Multijugador.bb") se encuentra el bucle principal del juego, el cual llama a las demás funciones. Pero antes, debemos preguntar al usuario si quiere ser el anfitrión o el invitado para que el programa ejecute una parte del código u otra. Este control lo realizamos simplemente a través de la variable global "Host" (1 ➠ Host / 0 ➠ Remoto).



Esquema de la relación entre un equipo remoto conectado a una partida creada por un equipo anfitrión (host).

Si se elige ser el host, se le pregunta al usuario qué tipo de juego va a crear y en qué escenario. Si por el contrario, decide ser el remoto, es decir, unirse a una partida creada, solo es necesario preguntarle el escenario donde se ha creado la partida y la dirección IP del host para poder entrar en contacto con él. Después de esto, el programa ya está preparado para diferenciar a ambos jugadores conectados. El siguiente paso es cargar modelos, texturas y el audio del juego, así como preparar el entorno y crear el resto de elementos. Seguidamente, preparamos el "stream" o fichero de comunicación para el envío de paquetes y creamos lo jugadores. Para esto último hay que tener un concepto claro: tanto el host como el remoto son jugadores locales; es decir, en el ordenador host, su jugador es el local y el jugador del ordenador invitado es el remoto. Y para el ordenador remoto, su



2 *La incursión de ovnis en la batalla por red incrementa la jugabilidad y la posibilidad de incluir un modo cooperativo.*

jugador es el local y el jugador del ordenador host es el remoto. Por lo tanto, en ambos casos debemos crear un jugador local. Ahora bien, su dirección IP y puerto de comunicación serán diferentes (Ver Código 1).

Cuando un jugador crea una partida (host) solo tiene que crear su bionave y colocarla en el terreno. Sin embargo, para el jugador remoto, además de crear su bionave, debe comunicárselo al host para que éste pueda crear y colocar en su terreno la bionave invitada. Así que bastará con enviarle un paquete con su nombre de identificación y su posición. Este paquete se identifica con el número 100. Veremos luego en el módulo de red cómo se recibe esta información por el otro equipo. Una vez preparada la comunicación, entramos en el bucle de juego donde controlamos y actualizamos al jugador local "Control_JugadorLocal(JugLocal)" y "Actualizar_JugadorLocal (JugLocal)", la comunicación "Actualizar_Red(JugLocal)" y el juego "Actualizar_Juego()". Las funciones para controlar al jugador local son exactamente igual a las utilizadas en el juego original de "Zone of Fighters". La única diferencia es que utilizamos como variables de control (Entidad, rotación y velocidad) los campos de la estructura "JugadorLocal" previamente definida en el módulo de definiciones:

```
Type JugadorLocal
     Field Entidad
     Field yy#,velocidad#
     Field Estado
End Type
```

(■) MÓDULO DE RED. CONTROLANDO LA COMUNICACIÓN

En este módulo controlamos todo el flujo de paquetes de información entre los dos juegos. Dependiendo de si el equipo se comporta como host o remoto entraremos en un control u otro. Sin embargo, las operaciones para los dos jugadores son básicamente las mismas pero orientadas a cada tipo de jugador.

(■) CONTROL POR EL HOST

Si hemos elegido ser el anfitrión del juego, el programa ejecutará las sentencias contenidas a partir de: "If Host". Lo primero que debemos hacer es obtener el paquete que nos envían por medio de "RecvUDPMsg" en la variable "Msg". A partir de ahí, podemos obtener la dirección IP y el puerto del jugador remoto. La IP ya la tenemos en "Msg" a través de "RecUDPMsg" y el puerto lo hallamos con "UDPMsgPort":

```
If Host
  Msg = RecvUDPMsg (UDP_Stream)
  IP_Remoto=Msg
  Puerto_Remoto=UDPMsgPort
  (UDP_Stream)
  ...
```

Lo siguiente es comprobar si realmente nos ha llegado un paquete. Si es así, obtenemos el primer byte que, como sabemos, nos dirá el tipo de información que trae (Tipo_Mensaje). De esta forma, podemos seleccionar las acciones por medio de una estructura "Select.. Case":

```
If Msg
  Tipo_Mensaje=ReadByte(UDP_Stream)
  Select Tipo_Mensaje
  ...
```

Código 1. La IP será diferente según quién elija ser el host

```
If Host
  IP = HostIP(1) :  IPSer$=DottedIP(IP) :  IP_host=Int_IP(IPSer$)
  Puerto_host=2001 :   Nombre_Host$=nombre$
  UDP_Stream=CreateUDPStream(Puerto_host)
  JugLocal.JugadorLocal= Crear_JugadorLocal ()
  PositionEntity JugLocal\Entidad,7300,300,800
Else
  Puerto_Remoto=Rand(3000,4000) :  Nombre_Remoto$=nombre$
  UDP_Stream=CreateUDPStream(Puerto_Remoto)
  JugLocal.JugadorLocal= Crear_JugadorLocal ()
  PositionEntity JugLocal\Entidad,7300,300,800
  WriteByte  UDP_Stream,100
  WriteLine  UDP_Stream,Nombre_Remoto$
  WriteShort UDP_Stream,7300
  WriteShort UDP_Stream,300
  WriteShort UDP_Stream,800
  SendUDPMsg UDP_Stream,IP_host,Puerto_host
EndIf
```



TRUCO

Una forma de ahorrar bytes en paquetes de información es evitar en lo máximo posible variables de tipo "Float", ya que ocupan 4 bytes cada una. Por ejemplo, si no precisamos de posiciones precisas, podemos sustituirlas por variables de tipo "Short" que ocupan la mitad. En nuestro caso, es lo mismo utilizar 500.5, 1200.8, 300 que 500,1200,300 ya que el terreno es de grandes dimensiones.

Como primera opción, vamos a gestionar el envío del paquete remoto de tipo 100 que hizo el jugador remoto al unirse a la partida (Ver Código 2).

Siguiendo la lectura del stream (recordemos que los datos en un stream se escriben y leen secuencialmente), encontramos el nombre del jugador remoto y su posición.

Evidentemente, si un jugador se ha unido a nuestra partida debemos verlo en la pantalla. Por lo tanto, es necesario crear otra bionave como jugador remoto además de la nuestra (jugador local).

El control de esta nueva bionave lo realizamos en el módulo "Juego.bb". Bien, hasta aquí todo perfecto. Sin embargo, si nos situamos en el ordenador del jugador nuevo veremos que él solo ve su bionave y no la del host. Así que éste debe enviar un paquete al remoto comunicándole que está dentro de la partida y que debe crear otra bionave que represente al host. Este paquete será de tipo 4 y se gestionará en la parte del código del remoto. El siguiente paso importante para el host es actualizar a su bionave invitada además de la suya propia. Obviamente, esta última es controlada por el jugador pero para la remota dependerá de la información que reciba del otro ordenador. Todos los paquetes de actualización recibidos por el remoto serán de tipo 2 y contendrá las coordenadas de posición X, Y,y Z de la bionave, así como dos bytes más indicando si ha disparado y con qué arma. Estos dos últimos datos son suficientes para que el programa anfitrión genere los disparos y explosiones con el código común.

```
Case 2
  RY#=ReadFloat(UDP_Stream)
  RPX=ReadShort(UDP_Stream):
  RPY=ReadShort(UDP_Stream):
  RPZ=ReadShort(UDP_Stream)
  Disp=ReadByte(UDP_Stream)
  Arma=ReadByte(UDP_Stream)
  Actualizar_JugadorRemoto
  (RY#,RPX,RPY,RPZ,Disp,Arma)
```

Debido a que el desplazamiento de la bionave y los disparos dependen de un pivote es necesario proporcionar el ángulo de rotación exacto al otro ordenador en una variable de tipo "Float"(RY#). Para actualizar esta bionave solo tenemos que posicionarla en las coordenadas recibidas, controlar su munición y la detección con los bonos. Estos dos últimos procesos se podían haber sustituido por variables contenidas en el paquete de información indicándolos. Pero, como dijimos, hay que evitar envíos sustituibles por otros métodos.

Los siguientes tipos de mensajes se refieren al control de la partida en sí como información si el jugador remoto ha muerto, abandona el juego o si la partida ha terminado.

Código 2. El jugador remoto envió un paquete de tipo 100

```
Case 100 ;Crear jugador remoto
  Nombre_Remoto$=ReadLine(UDP_Stream)
  RPX%=ReadShort(UDP_Stream):RPY%=ReadShort(UDP_Stream):
  RPZ%=ReadShort(UDP_Stream)
  JugRemoto.JugadorRemoto=Crear_JugadorRemoto()
  PositionEntity JugRemoto\Entidad,RPX,RPY,RPZ
  WriteByte  UDP_Stream, 4
  WriteLine  UDP_Stream,Nombre_Host$
  WriteShort UDP_Stream,EntityX(JugLocal\Entidad)
  WriteShort UDP_Stream,EntityY(JugLocal\Entidad)
  WriteShort UDP_Stream,EntityZ(JugLocal\Entidad)
  WriteByte  UDP_Stream,Modo_Juego
  WriteByte  UDP_Stream,NumOvnis
  SendUDPMsg UDP_Stream,IP_Remoto,Puerto_Remoto
```



3

El nuevo módulo para multijugador incluye dos tipos de ambientación diferente.

Es conveniente enviar toda la información posible de una sola vez en un solo paquete que muchos envíos de pequeños paquetes. Por eso, implementamos en este mismo módulo el envío de información después de analizar los paquetes recibidos. Aun siendo el host y jugador local en nuestro equipo, en el ordenador del remoto nos corresponde ser el remoto y debemos mandarle nuestra posición y estado continuamente antes de terminar y actualizar el juego (Ver Código 3).

(■) CONTROL POR EL JUGADOR REMOTO

Si hemos elegido formar parte de una nueva partida, nos convertiremos en un jugador remoto. Por lo tanto, el programa ejecutará el código a partir de la instrucción "Else" de la función "Actualizar_Red (JugLocal.JugadorLocal)".

Al igual que ocurría con el host, el remoto también recibe y procesa información proveniente de éste con "RecvUDPMsg". Los tipos de mensajes son iguales pero con otro número de identificación diferente. Cuando nos unimos a un host debemos crear su bionave en nuestro terreno. Para ello, debemos haber recibido un mensaje de tipo 4 enviado por el host cuando entramos en la partida:

```
Case 4 ; Crear jugador Host
  Nombre_Host$=ReadLine (UDP_Stream)
```

```
  RPX=ReadShort(UDP_Stream):
RPY=ReadShort(UDP_Stream):
RPZ=ReadShort(UDP_Stream)
  Modo_Juego=ReadByte(UDP_Stream)
  NumOvnis=ReadByte(UDP_Stream)
  JugRemoto.JugadorRemoto=
  Crear_JugadorRemoto()
  PositionEntity JugRemoto\
  Entidad,RPX,RPY,RPZ
  For n= 1 To NumOvnis
    Crear_OVNI()
  Next
```

Como podemos observar, hemos recibido también información del juego creado ("Modo_Juego" y, "NumOvnis") y que necesitaremos para controlar la partida en el remoto, ya que éste carece de esa información.

Si recordamos, el host envía continuamente paquetes de tipo 1 con su posición y estado al jugador remoto para actualizar su posición. Por lo tanto, debemos tener en este caso el mismo código del host:

```
Case 1 ; Actualiza Host
  RY#=ReadFloat(UDP_Stream)
  RPX=ReadShort(UDP_Stream):
  RPY=ReadShort(UDP_Stream):
  RPZ=ReadShort(UDP_Stream)
  Disp=ReadByte(UDP_Stream)
  Arma=ReadByte(UDP_Stream)
  Actualizar_JugadorRemoto
  (RY#,RPX,RPY,RPZ,Disp,Arma)
```

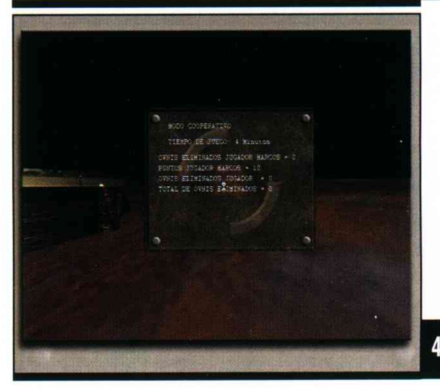
Es importante que esté claro que esta información solo la utiliza el jugador remoto para actualizar la bionave que representa al host en su terreno de juego. Para terminar, al igual que ocurre con el host, es necesario que el remoto le envíe continuamente su posición y estado antes de salir de la función. En este caso en un paquete de tipo 2.

(■) LOS OVNIS

La presencia de ovnis en el juego conlleva un envío de información para que en ambos equipos se muestren en la misma posición y con el mismo estado durante la partida. Esto trae consigo un envío adicional muy importante de información, la cual debemos realizar junto a la de los jugadores en el módulo de red. Luego, dependiendo de si el programa es host o remoto, se ejecutará una función de actualización diferente para los ovnis. Para disponer de las coordenadas y estado de cada ovni de una forma ordenada y que podamos enviarlo todo junto en el módulo de red, vamos a disponer de una estructura de datos temporal que almacene estos valores en la función "Actualizar_OVNIS_Host ()" del módulo "Ovnis.bb":

```
Temp.tipo_temp = First tipo_temp
  For OVNI.tipo_OVNI= Each
  tipo_OVNI
    ...
    Temp\OVNIX#=OVNI\x
    Temp\OVNIY#=OVNI\y
    Temp\OVNIZ#=OVNI\z
    Temp\vida=OVNI\vida
    Temp\Disparo=OVNI\
    Send_disparo_OVNI
    Temp\Arma=OVNI\tipo_armamento
    Temp = After Temp
Next
```

Seguidamente, los datos de esta estructura los volcamos en el stream de envío, los cuales serán leídos por el remoto y



Es importante mostrar una estadística de resultados del juego al final de la partida, ya que aumenta la competitividad entre los jugadores, lo cual invita a seguir jugando.

que utilizará la función "Actualizar_OVNIS_remotos ()" del módulo "Ovnis.bb":

```
For Temp.tipo_temp = Each tipo_temp
  WriteShort UDP_Stream,Temp\OVNIX#:
  WriteShort UDP_Stream,Temp\OVNIY#
  WriteShort UDP_Stream,Temp\OVNIZ#:
  WriteByte UDP_Stream,Temp\vida
  WriteByte UDP_Stream,Temp\Disparo:
  WriteByte UDP_Stream,Temp\Arma
  WriteByte UDP_Stream,Temp\objetivo
Next
```

Hay que recordar que el host es quien crea y desplaza a los ovnis y el remoto solo se limita a tenerlos como clones en el terreno.

Los procesos expuestos en este número son los básicos en este nuevo módulo para "Zone of Fighters". El resto del programa está ampliamente comentado en el código fuente, incluido en el CD. Siguiendo el flujo del programa y leyendo los comentarios es fácil comprender su funcionamiento.

... Hasta la próxima.

Código 3. Debemos mandar continuamente nuestra posición y estado

```
If JugLocal\estado=1
  WriteByte  UDP_Stream,1
    Else
  WriteByte  UDP_Stream,10 ; Muerto
EndIf
WriteFloat UDP_Stream,JugLocal\yy :          WriteShort UDP_Stream,EntityX(JugLocal\Entidad)
WriteShort UDP_Stream,EntityY(JugLocal\Entidad) :  WriteShort UDP_Stream,EntityZ(JugLocal\Entidad)
WriteByte  UDP_Stream,Send_disparo :         WriteByte  UDP_Stream,tipo_armamento
SendUDPMsg UDP_Stream,IP_Remoto,Puerto_Remoto
```

Técnicas avanzadas de diseño **gráfico**

Para finalizar la sección de diseño gráfico hemos querido recorrer algunas de las herramientas que hemos utilizado para avanzar un poco más en su uso, aprendiendo algunas técnicas avanzadas que nos permitirán alcanzar mayores cotas de calidad en nuestros trabajos.

TÉCNICAS CON DEEP PAINT 3D

Prácticamente, Deep Paint 3D tiene las características de un programa de diseño gráfico como herramientas de dibujo lineal y vectorial, además de trabajar con texto. También, Deep Paint 3D admite la posibilidad de trabajar con capas como si de un programa de diseño como Paint Shop Pro o Adobe PhotoShop se tratara (de hecho está basado en el funcionamiento de Photoshop).

TEXTO SOBRE LA TEXTURA

En nuestro primer ejercicio vamos a utilizar la bionave de "Zone of Fighters" para colocarle en una de las alas la frase "ZOF-Mod X". Ejecutamos el programa y cargamos el modelo "bionave.obj"contenido en el directorio "Extras" del CD. Pulsamos directamente en "OK" en la ventana *Import Material* del comienzo. Aparecerá el modelo de espaldas y sin textura. Para cargar la textura pulsamos F7 para ir a la sección de capas en el panel de comandos. Hacemos clic en la casilla "C" del color y elegimos la casilla *An Image*. Elegimos del disco la textura de la bionave "textura_bionave". El modelo aparecerá texturizado. El siguiente paso es situar en la ventana a la bionave mostrando la parte superior de las alas (Fig. 1).

Podemos mostrar la textura en una nueva ventana haciendo doble clic sobre la casilla del color "C" en la capa. Lo que hagamos en la textura se mostrará sobre el modelo y viceversa. Con la lupa englobamos las dos alas de la textura y elegimos la herramienta de texto [T]. Se abrirá una ventana donde podemos escribir el texto con la fuente que queramos. Al pulsar "OK" aparecerá un recuadro con la frase que hemos escrito sobre la textura (Fig. 2).

Haciendo clic y desplazando en los bordes del recuadro podemos cambiar el tamaño de la frase o bien en la ventana de opciones *Move Options* en las casillas *Width* y *Heigth*. Para finalizar colocamos la frase en el sitio correcto sobre una de las alas. Observamos cómo se puede ver directamente sobre el modelo en la ventana 3D. Una vez colocada podemos aceptar pulsando de nuevo sobre la herramienta de texto (Fig. 3).

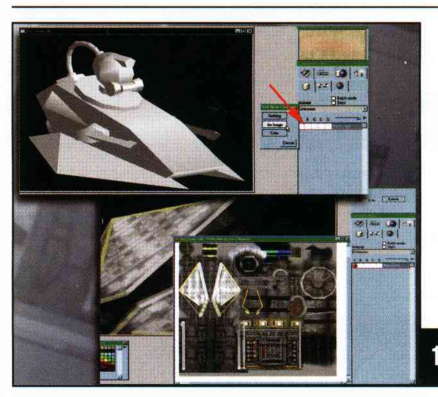
TRABAJAR CON CAPAS

Podemos disponer de más de una capa con todos los atributos (color, bump, glow, shini-

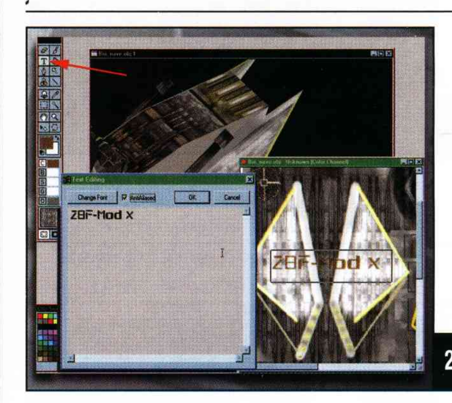


RECORDATORIO

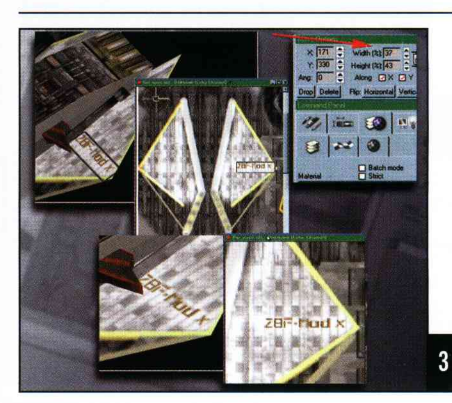
Para girar el modelo con el ratón en la ventana debemos dejar pulsada la tecla "R" y para desplazarlo la tecla "ESPACIO". El zoom se activa con la tecla "Z". Para aumentar, hacer clic mientras mantenemos pulsada la tecla "Control" y para alejarnos con la tecla "Alt".



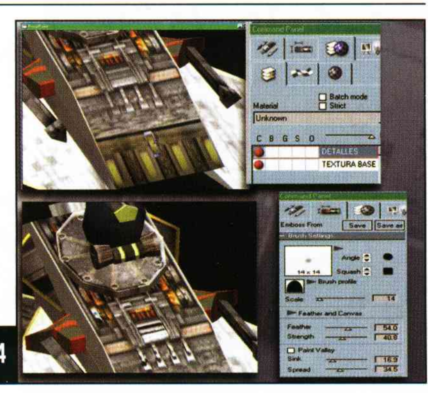
Es posible obtener la textura en una nueva ventana para realizar cambios en 2D.

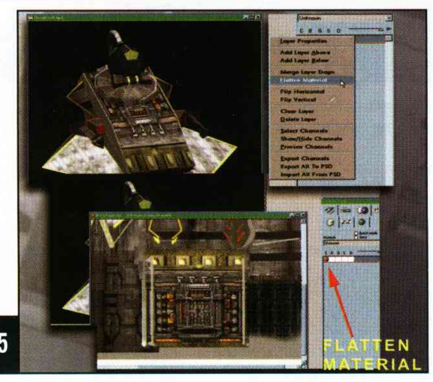


Con la herramienta de texto podemos incorporar palabras a la textura directamente.

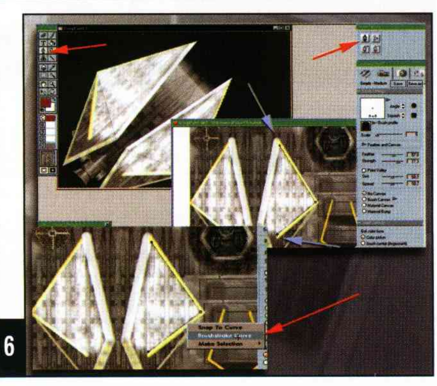


El ajuste del texto se realiza por igual en 2D y 3D directamente en el modelo.


4
Trabajando con nuevas capas podemos aplicar multitud de efectos sin alterar la textura original.



5
Una vez que hayamos terminado de trabajar con las capas, debemos unirlas todas para obtener la textura final.


6
Por medio de los splines podemos dibujar formas complejas con precisión.

ness y opacity). Para crearla solo tenemos que hacer clic con el botón derecho sobre la sección de capas (F7) y elegir la opción *Add Layer*. Esta nueva capa puede contener otra textura (la cual podemos mezclar) o bien utilizarla para pintar sobre la textura o aplicar algún tipo de efecto sin afectar a la textura original. Precisamente, vamos a realizar un ejercicio para aplicar en algunas partes de la textura un efecto *emboss* para realzar la profundidad de la textura. Para ello, seleccionamos la casilla "C" de la nueva capa y seleccionamos la opción *Nothing*. Ya tenemos preparada la nueva capa vacía. Elegimos un pincel de punta cuadrada (F6) en *Brush Settings* con un tamaño de *Feather* de 54 y un *Strength* de 40. Pasamos a la sección *Presets* (F5) y elegimos el efecto *Emboss From Layer Below* de la lista *Image Processing*. Sobre algunas partes delanteras del modelo hacemos clic con el pincel. Observamos cómo la textura parece levantarse. Podemos realizar la misma operación con un pincel de punta redonda sobre los tornillos de la base de la cabeza (Fig. 4).

Todos estos efectos los hemos aplicado en la nueva capa, en ningún momento hemos modificado la textura original. Si desplazamos el deslizador de opacidad de la capa (junto a la casilla "O") hacia la izquierda veremos cómo el efecto desaparece. Siempre podemos borrar la nueva capa haciendo clic con el botón derecho del ratón sobre el nombre de la capa y eligiendo la opción *Delete Layer*, o bien *Clear Layer* para borrar solo el efecto. Podemos disponer de cuantas capas queramos y podemos realizar cualquier operación desde pintar hasta mezclar otras imágenes. Si después de trabajar con varias capas sobre la textura (aplicando efectos o pintando) queremos salvar la textura con todo en un archivo, debemos unir todas las capas en una con la opción *Flatten Material* del menú emergente que aparece al hacer doble clic sobre el nombre de la capa (Fig. 5).

TRABAJAR CON *SPLINES*

Los *Splines* son curvas vectoriales similares a las utilizadas en los programas de dibujo. Deep Paint 3D admite su uso para poder dibujar curvas o siluetas variadas que difícilmente se harían a mano alzada. Como ejercicio, vamos a pintar el borde del ala de la bionave de color rojo. Para ello vamos a utilizar *Splines*. Seleccionamos la herramienta de la pluma. Pulsamos sobre el primer icono que aparece en la ventana de opciones *Spline Options* situada sobre el panel de comandos. Este icono (punta de pluma) sirve para colocar puntos de unión, el icono situado debajo (pluma con un +) sirve para insertar puntos y el de la derecha para borrar (pluma con un -). El segundo icono sirve para convertir un punto en una curva editable. Con el primer icono pulsado hacemos clic sobre la punta del ala (en la propia textura), luego hacemos clic en la curva y para finalizar un último clic junto al final de la línea amarilla de la textura. Vemos cómo los tres puntos están unidos por una línea. Elegimos una herramienta de pincel con punta cuadrada con tamaño (*Scale*) de 4 y elegimos un color rojo para la tinta. A continuación para crear el dibujo de la línea hacemos clic con el botón derecho sobre la línea del *Spline* y elegimos la opción del menú emergente *BrushStroke Curve* (Fig. 6).

Vemos cómo el *Spline* se ha convertido en una línea gruesa pintada de color rojo. Estas operaciones siempre son convenientes hacerlas en una capa nueva. A continuación, vamos a realizar la misma operación en la otra ala, pero en este caso, realizaremos una línea curva. Para ello hacemos clic en el extremo del ala y luego otro clic

en el otro extremo y sin dejar de pulsar desplazamos el ratón. Veremos cómo el último punto va formando una curva con relación al primero (Fig. 7).

OPCIONES DE RENDERIZADO Y LUCES

Para poder tener una referencia de cómo se verá el modelo texturizado con un motor 3D, Deep Paint 3D permite modificar su representación en la ventana. Es posible definir la calidad de dibujado seleccionando qué atributos queremos ver, el fondo de la ventana y modificar la intensidad y posición de la luz ambiental y puntual. Estas opciones están situadas en la sección *Settings* (F8) del panel de comandos. En este apartado encontramos la pestaña *Lighting* para las luces y *Scene* para el render. Situándonos en *Lighting* vemos que podemos modificar la posición del foco de luz pulsando sobre los focos que rodean a la esfera, la cual representa el modelo. En el deslizador *Ambient* modificamos la intensidad de luz general, mientras que en *Spot* la del foco. Pasemos a la zona *Scene*. Ahí encontramos dos niveles de render o dibujado: uno cuando el modelo está estático *(Still)* y otro cuando lo movemos con el ratón (*Moving*). Pulsando sobre el icono de la esfera marrón modificamos la representación: con textura, en modo *flat* (sin textura) y en alámbrico (*wireframe*). Además, podemos activar o desactivar cada uno de los atributos de la capa en ambos casos. Para cambiar el color del fondo de la ventana, elegimos el color y pulsamos en el botón "Color" (Fig. 8).

TÉCNICAS CON LITHUNWRAP

En ocasiones puede ocurrir que la textura no coincida con el despiece del modelo sobre la plantilla UV. Así que vamos a aprender cómo realizar este ajuste con LithUnwrap.

En primer lugar, cargamos el modelo "bionave.obj" en *File / Import*, el cual ya contiene un mapeado UV. Seguidamente, la textura "textura_Bionave" en *File / Material / Open*. Si activamos la ventana de previsualización del modelo en *Preview / Show Model* observamos con detenimiento que la textura está desplazada. Por lo tanto, tenemos que ajustar la plantilla a la textura. Para el ejercicio, ajustaremos un par de partes. La primera de ellas será la base del cuello (Ver Fig. 9).

Para seleccionar los vértices elegimos la opción *Select / Vertex*. Con el ratón hacemos un recuadro que englobe a los cuatros vértices inferiores de la base en la plantilla.

Sin dejar de hacer clic sobre cualquiera de los vértices lo movemos hasta que queden ajustados.

Hagamos lo mismo para parte de la cubierta superior de la nave. Para esta ocasión seleccionamos la opción *Select / Group*, ya que vamos a elegir toda una pieza completa del despiece. Para seleccionarla bastará con hacer clic sobre cualquiera de sus vértices.

Con mucha paciencia se puede conseguir un ajuste adecuado que seguro mejorará notablemente el aspecto de nuestro modelo.

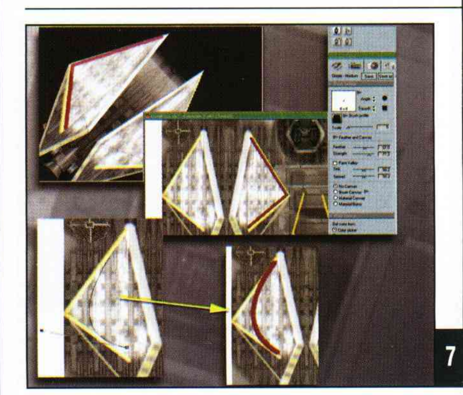
TRABAJANDO CON LAS PROPIEDADES DE LAS CAPAS EN PAINT SHOP PRO

No queremos terminar sin realizar algunos ejercicios interesantes con Paint Shop Pro. En



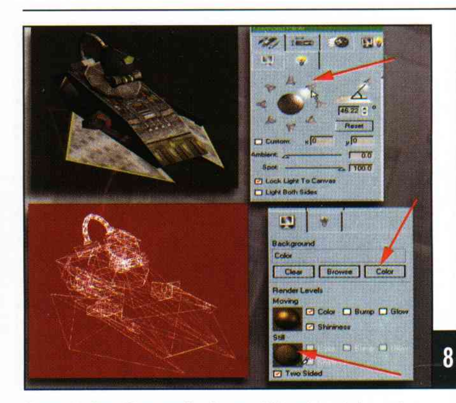
TRUCO

Para ver mejor el ajuste de la textura en el modelo, podemos abrir una ventana de previsualización. Observaremos que a medida que movemos la plantilla se va ajustando en tiempo real en el modelo previsualizado.



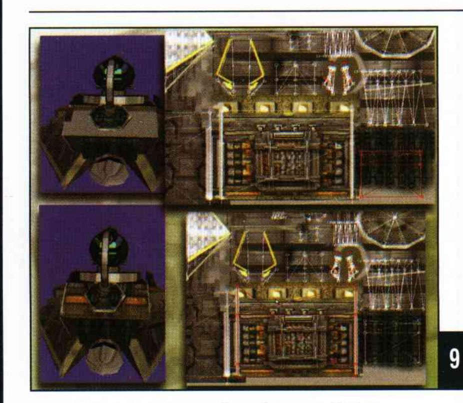
7

Para crear formas curvas perfectas podemos ir sumando puntos al spline.



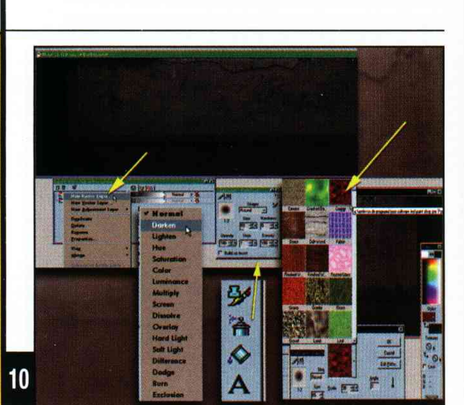
8

Deep Paint 3D permite la modificación del render del modelo en 3D y de la iluminación.



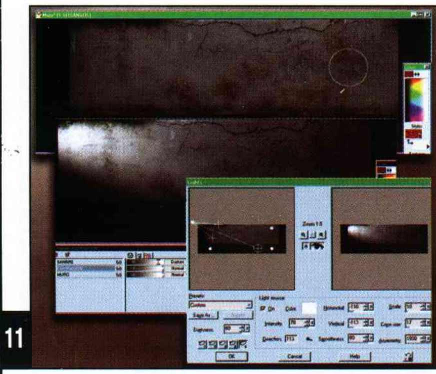
9

El ajuste de la textura al modelo con LithUnwrap resulta sumamente sencillo y proporciona unos resultados fantásticos.



10

Trabajar con las propiedades de las capas en Paint Shop Pro nos permite simular cualquier aspecto para una textura.



11

Aprovechando los efectos de iluminación podemos personalizar la textura para adaptarla a un ambiente determinado.



12

Aplicando la propiedad Ligthen conseguimos visualizar todos los píxeles más claros de la mezcla y los colores base.

primer lugar queremos completar una textura de un muro conseguida a partir de técnicas fotográficas. La idea es añadir a la pared manchas de sangre y conseguir, por medio de iluminación, que la pared forme parte de un espacio interior ligeramente iluminado por una ventana. En este ejercicio vamos a trabajar con las propiedades de las capas. Comenzamos, cargando la imagen del muro contenida en "Extras" del CD "muro.png". Para conseguir las manchas de sangre sería erróneo pintar directamente sobre la textura, así que utilizaremos una nueva capa. Para crearla, hacemos clic con el botón derecho del ratón sobre la capa del fondo. En el menú emergente elegimos la opción *New Raster Layer*. A esta nueva capa la llamamos "Sangre". Para manchar la pared vamos a utilizar la herramienta del bote de spray. En la ventana de opciones cambiamos el tamaño del pincel redondo a 65. En la casilla *Hardness* colocamos un 0 y en *Step* un 1, con esto conseguimos que el spray pinte con menos frecuencia, logrando así manchas discontinuas. Para conseguir el aspecto elegimos de la paleta *Colors* el estilo *Craggy* de la librería para la tinta. Pasamos entonces a pintar las manchas sobre el muro en la nueva capa. Una vez terminado el proceso pinchamos en la flechita situada junto a la palabra "Normal". Emergerá un menú con todas las propiedades de mezclado de la capa: *Layer Blend Mode* (Fig. 10).

Estas propiedades determinan la forma de mezclar los píxeles de la capa activa con los de la capa situada debajo. Realmente esta operación no combina las capas, sino que muestra cómo se combinan. Disponemos de 17 formas de combinar las capas entre ellas *Hue* (tono), *Color*, *Saturation* (Saturación) y *Luminance* (Luminancia); solo funcionan para imágenes de 24 bits. Para nuestro ejemplo, seleccionamos la mezcla *Darken*que elimina los píxeles más claros de la capa de abajo. Además, vamos a reducir la opacidad de la capa hasta un 85% (Fig. 11).

El útltimo paso es crear el efecto de iluminación. Para ello, duplicamos la capa del fondo y la llamamos "Iluminación". Seleccionamos el efecto *3D effects / Illuminations effects / Lights*. Aparece una ventana de diálogo con dos ventanas. En la de la izquierda situamos los diferentes focos de luz y en la derecha vemos un previo de los resultados obtenidos. Para nuestro ejercicio elegimos unos focos estándar (customs) de la casilla *Presets*. Se nos muestran cinco focos de luz representados por puntos blancos. Al pinchar sobre cualquiera de ellos se muestra su ángulo de incidencia y su radio de acción en el centro. Eliminamos la luz de la esquina superior izquierda deseleccionando la casilla *on* en *Light source*. La quinta luz la situamos como se muestra en la figura y seleccionamos los siguientes parámetros:

- **En Color** elegimos el blanco puro con una intensidad (*Intensity*) del 78%.
- **En Darkness** colocamos un 69.
- **En Direction** colocamos 113.

Esperamos que estos pequeños ejercicios hayan servido para mejorar aún más el conocimiento de estas estupendas herramientas de desarrollo. El tesón y la práctica diaria enseñan el resto.

El mundo del diseño gráfico está lleno de herramientas más o menos potentes. Pero siempre se encuentra, en cada una de ellas, una opción diferente que nos ayuda a conseguir el resultado que andamos buscando. La especialización es cada vez más abundante y solamente un 5% de las herramientas del mercado son las más utilizadas profesionalmente. Sin embargo, siempre es bueno conocer un poco de todo para estar preparados.

Secuenciadores trackers (y III)

En el número anterior empezamos a desarrollar un pequeño ejemplo con ModPlug Tracker.

Continuamos, en esta ocasión, completando el patrón rítmico. Seguidamente, añadiremos alguna melodía, efectos y trabajaremos con la edición de patrones.

EDITANDO EL PATRÓN II

Siguiendo con el patrón del número anterior, añadamos en el canal 3 algunos toques de maracas para completar el ritmo base. Colocando, más o menos de forma aparentemente desordenada, las notas de maracas conseguimos romper la monotoneidad del ritmo impuesto por el bombo y la caja de los canales 1 y 2. Recordemos que el sonido de las maracas corresponde al número 3 de las muestras cargadas. Nos situamos al principio del patrón en el canal 3 y disponemos las notas más o menos como se muestra en la figura 1.

Ya tenemos terminado el primer patrón con el ritmo base. Para reproducirlo cíclicamente podemos hacer "CTRL" + "F7".

Pasemos a continuación a completar el patrón, colocando las notas que servirán de melodía básica en el canal 4.

Para ello, elegimos el sonido número 4 "Vangelis" y colocamos las siguientes notas:

- DO equivalente a la tecla Q (C4) en la posición 0
- DO# equivalente a la tecla W (C#4) en la posición 24
- RE # equivalente a la tecla R (D#4) en la posición 30
- Tecla Q en 32 y en la posición 55, B3. Haciendo doble clic en esta posición y eligiendo del menú flotante la nota.

Es importante tener en cuenta que la muestra sonará completa la primera vez que se reproduzca. Cando termine, volverá a reproducirse comenzando por la posición indicada en la casilla *Start* en *Loop* dentro de la sección de muestras *Samples*. Si en esta casilla colocamos un 0 indicaremos que la muestra sonará completa cíclicamente. La posibilidad de cambiar el comienzo del bucle en una muestra nos permitirá mantener su reproducción hasta la llegada de otra nota.

Para terminar el patrón necesitamos añadir dos canales más. Para ello, solo tenemos que ir a la pestaña *General* y cambiar el *Song Type* a *6 Channels* pulsando en el botón "Change". Seguidamente, añadamos un nuevo sonido a la lista. Pulsamos



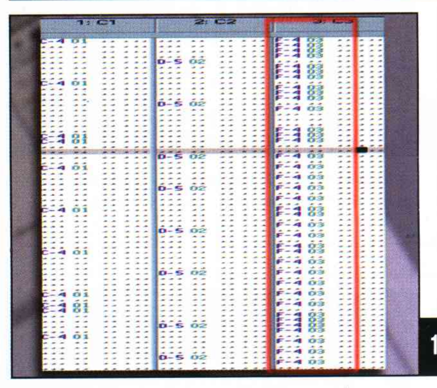
NOTA

Para reproducir la canción completa desde el principio, es decir, todos los patrones, basta con pulsar F5.



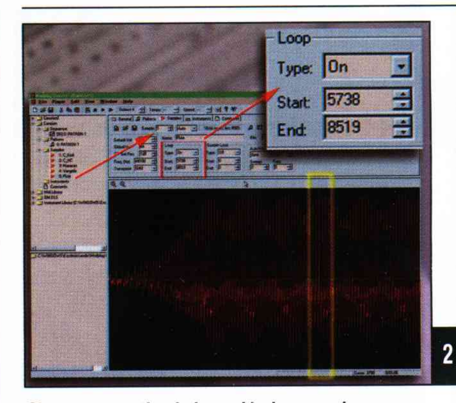
NOTA

Podemos seleccionar cada parte de la canción (patrones, secuencias o sonidos) desde el explorador de proyectos situado a la izquierda del programa. Basta con desplegar cada carpeta y seleccionar con el ratón. Por ejemplo, para elegir la muestra 4 "Vangelis" hacemos clic en el signo + junto a la carpeta "Samples" y posteriormente sobre el sonido a elegir. Veremos cómo en la casilla *Instrument* del editor aparece el sonido elegido.



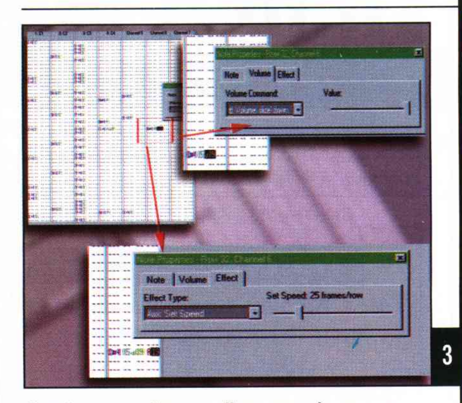
1

Colocando las notas de maracas de forma más o menos desordenada se consigue un ritmo menos repetitivo.



2

Si tenemos activada la opción Loop podemos modificar la reproducción continua de la muestra en toda la canción.

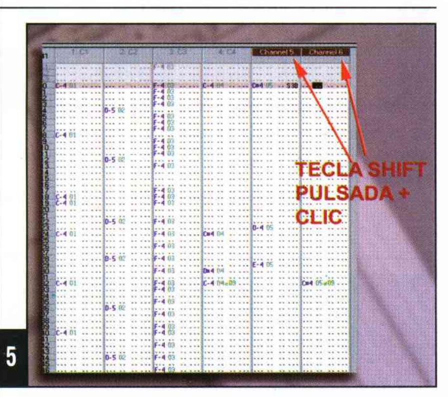


3

A cada nota podemos aplicar operaciones a su volumen y añadir efectos.

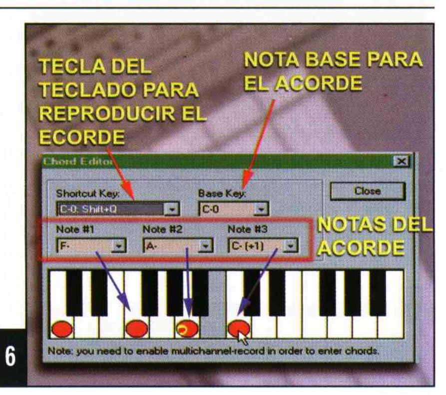
4

A través del selector de patrones podemos duplicar, borrar o insertar patrones en la canción.



5

Con ModPlug podemos grabar simultáneamente en varios canales a la vez.



6

La posibilidad de utilizar una combinación de teclas del ordenador para producir acordes nos ayudará a simplificar el trabajo.

TRUCO

Para borrar una muestra de la lista, hacemos clic con el botón derecho sobre su nombre en el explorador de proyectos y elegimos la opción *Delete sample*.

en la pestaña *Samples* y cargamos la muestra "Flute". Al reproducirla oímos cada vez que vuelve al comienzo. Para conseguir el audio sostenido, es decir, sin cortes, debemos colocar el comienzo del loop en una posición más adelantada. En este caso colocamos en la casilla *Start* de *Loop* el 5738. Si reproducimos ahora el sonido no tendrá cortes (Fig. 2).

Volvamos a nuestro patrón y coloquemos las notas del nuevo sonido en las siguientes posiciones en el canal 5: C#4 Pos. 0, D4 Pos. 23, E4 Pos. 29 y C4 en posición 55. Podemos oír cómo hemos conseguido que la flauta se quede sonando hasta que encuentre otra nota.

ALGUNOS EFECTOS

Podemos aprovechar esta cualidad para realizar algunos matices de armonía. Por ejemplo, coloquemos una sola nota de flauta en el canal 6, concretamente C#4 en la posición 32. Al reproducir el patrón oímos cómo esta nota solitaria se mantiene sonando en todo momento. El inconveniente de esto es que rompe la armonía que hemos creado. Por ejemplo, queremos que siga sonando solo hasta el final del patrón. Para lograrlo, podemos aplicar una bajada de su volumen. Los procesos de volumen los encontramos en el menú de propiedades de la nota (aparece haciendo doble clic sobre la nota) en la pestaña *Volume*. Elegimos *d: Volume Slide Down* con el valor máximo. Este efecto bajará el volumen de la nota progresivamente.

Otro tipo de efectos se halla en la pestaña *Effects*. Encontramos efectos para cada nota y otros que afectarán al conjunto del patrón. Por ejemplo, si queremos modificar la velocidad de reproducción a partir de cierta nota solo tenemos que elegir el efecto *Axx. Set Speed*. En la sección *General* también podemos aplicar efectos a cada canal completo en la casilla *Effect* (Fig. 3).

OPERACIONES ESPECIALES

Una vez que hemos terminado el primer patrón podemos duplicarlo cuantas veces queramos. Para ello, hacemos clic con el botón derecho sobre el patrón 0 del selector de patrones y elegimos la opción *Duplicate Patterns* (Fig. 4).

Como podemos comprobar, además de esta operación, en el mismo menú podemos crear, insertar, borrar y copiar patrones. Otro procedimiento que puede resultar útil es la grabación de varios canales al mismo tiempo. Cada vez que coloquemos una nota el cursor avanzará y se colocará en el otro canal y así sucesivamente. Para realizar esta operación tenemos que activar los canales que queremos grabar haciendo clic sobre el nombre del canal mientras mantenemos la tecla "SHIFT" pulsada, al hacerlo el nombre cambiará de color indicando que está activado el proceso (Fig. 5).

Por último, una operación muy interesante es la grabación de acordes a partir de combinaciones de teclas. Para poder utilizar este procedimiento debemos activar con anterioridad los canales que recibirán cada una de las notas del acorde. Una vez activados pulsamos en el icono para abrir el editor de acordes *Chord Editor*. Podemos tener hasta tres notas por acorde representadas en el teclado por un círculo rojo. Los acordes se forman a partir de una nota base, la cual situamos desde la casilla *Base Key*. Para elegir las notas del acorde es suficiente con hacer clic en el teclado. En la casilla *Shortcut Key* elegimos la combinación de teclas que producirá el acorde. Observamos cómo al pulsar la combinación que hemos seleccionado se graban en los canales elegidos previamente las notas del acorde en la misma posición (Fig. 6).

... Esperamos que todas las herramientas expuestas en este curso sirvan para facilitar el desarrollo del sonido que vuestros juegos merecen.

Funciones de red (y III) y ampliación del lenguaje

No queremos dejar las funciones de red e internet sin antes haber realizado algunos ejercicios para conocer mejor su funcionamiento. También damos a conocer algunos trucos que nos ayudarán a optimizar nuestro código y que mostramos en las figuras 1, 2 y 3.

Por último, hablaremos de las posibilidades de ampliación del lenguaje y de su futuro próximo.

CHATEANDO POR UDP

En este primer ejemplo hemos querido realizar un sencillísimo programa que permite hablar entre dos ordenadores conectados por red utilizando el protocolo UDP. El fundamento es que el mismo programa envíe y reciba mensajes utilizando un puerto de entrada y salida y otro de destino, el cual equivale al de entrada. Además, utilizamos dos streams para albergar los mensajes de envío y llegada. Cada ordenador deberá introducir la dirección IP del otro para establecer la comunicación.

Al principio del programa se pide la IP del equipo local para obtener información. Esta IP se visualizará como título de la ventana:

```
ContIP=CountHostIPs(GetEnv
("LocalHost"))
For n=1 To ContIP
  IPLocal$=IPLocal$+DottedIP
  (HostIP(n))
Next
AppTitle "IP Local = "+IPLocal
```

Para obtener la IP local utilizamos la instrucción "CountHostIPs". La instrucción "GetEnv" nos proporcionará el nombre del equipo en la red. Después de obtener la notación de la IP preguntamos a qué dirección IP vamos a comunicarnos. Así que la pedimos en notación 1.2.3.4 y luego la convertimos a la notación de número entero que entiende Blitz3D:

```
Locate 10,20:IPDest$=Input$
("IP Destino? ")
IPDestino= IntIP(IPDest$)
```

La función "IntIP" simplemente transforma el string con notación 1.2.3.4 a un número entero (Ver Código 1).

Para ello, aislamos los cuatro números separados por un punto y sumamos sus valores previamente calculados con la función "Val":

```
Function Val (String$)
  valor=0
  If Left(Trim(String$),1)="-"
    signo=-1
  Else
    signo=1
  EndIf
  For n=1 To Len(String$)
  c=Asc(Mid$(String$,n,1))
     If c>47 And c<58 Then
       m = valor :   m=m Shl 1:
       valor=valor Shl 3
       valor=valor+m+c-48
     EndIf
     If c=46 Exit
   Next
   valor=valor*signo
   Return valor
End Function
```

Código 1. Función "IntIP"

```
Function IntIP (IP$)
a1=val(Left(IP$,Instr(IP$,".")-1)):IP$=Right(IP$,Len(IP$)-Instr(IP$,"."))
a2=val(Left(IP$,Instr(IP$,".")-1)):IP$=Right(IP$,Len(IP$)-Instr(IP$,"."))
a3=val(Left(IP$,Instr(IP$,".")-1)):IP$=Right(IP$,Len(IP$)-Instr(IP$,"."))
a4=val(IP$)
Return (a1 Shl 24) + (a2 Shl 16) + (a3 Shl 8 ) +a4
End Function
```

Esta función simplemente convierte un string en un número con signo incluido, recorriendo cada uno de sus caracteres y convirtiéndolo con la instrucción "Asc".

Seguidamente, pasamos a definir los puertos y a crear los streams de entrada y salida.

```
PuertoEntrada=6000
PuertoSalida   =6300
PuertoDestino=6000
UDP_StreamR=CreateUDPStream
(PuertoEntrada)
UDP_StreamW=CreateUDPStream
(PuertoSalida)
```

Ya tenemos el programa preparado para enviar y recibir mensajes. Por lo tanto, ya podemos entrar en el bucle principal del programa. La idea es que a medida que escribamos los caracteres se imprimirán en el otro ordenador, admitiendo también el cambio de línea con la tecla "Enter". Para ello utilizamos la función "GetKey()" para obtener los caracteres desde el teclado uno a uno. A medida que tecleamos enviaremos cada carácter al otro equipo a través del stream de salida "UDP_StreamW". Igual que los caracteres, enviamos el cambio de línea, el cual corresponde al carácter 13:

```
Tecla=GetKey()
  If Tecla>0
    If Tecla=13
      Print
```

```
    WriteByte(UDP_StreamW,
    Tecla)
    SendUDPMsg UDP_StreamW,
    IPDestino,PuertoDestino
  Else
    Write Chr(Tecla)
    WriteByte(UDP_StreamW,
    Tecla)
      SendUDPMsg UDP_StreamW,
      IPDestino,PuertoDestino
  EndIf
EndIf
```

Por último, nos queda recibir el mensaje del otro ordenador. Para ello, utilizamos la instrucción "RecUDPMsg" y el stream de lectura "UDP_StreamR":

```
Mensaje=RecvUDPMsg(UDP_StreamR)
  If Mensaje<>0
    Longitud=ReadAvail
    (UDP_StreamR)
    If Longitud>0
      Caracter=ReadByte
      (UDP_StreamR)
      If Caracter=13
        Print
      Else
        Write Chr(Caracter)
      EndIf
    EndIf
EndIf
```

Antes de leer e imprimir el carácter en pantalla comprobamos que hemos recibido un mensaje correcto a través de su longitud.

Evidentemente, para probar este ejemplo debemos tener dos equipos conectados entre sí por cable de red.

INTERNET

Queremos mostrar unos pequeños ejemplos de comunicación a través de internet utilizando el protocolo TCP/IP.

ENVIAR UN EMAIL

En primer lugar, mostramos la forma más sencilla de enviar un email a una dirección de Internet utilizando la función "OpenTCPStream". Para poder establecer la comunicación debemos introducir la dirección donde queremos enviar el email. En este caso es admisible la notación "www" o bien "1.2.3.4":

```
Direccion$=Input$ ("DIRECCION
(notacion IP o www)? ")
Texto=Input$("Escriba el Email:")
```

Una vez introducido el texto que queremos enviar a la dirección especificada, abrimos el archivo de escritura con "OpenTCPStream":

```
Email=OpenTCPStream (Direccion$, 80)
If Not Email RuntimeError
("ERROR EN LA CONEXIÓN")
```

Si la operación devuelve un valor nulo salimos del programa con un error. Si la comunicación se ha inicializado correctamente pasamos a mandar el texto escribiendo el stream abierto.

```
WriteLine Email,Texto
Print "EMAIL: "
Print "<" + Texto+" >"
Print "ENVIADO CORRECTAMENTE"
```

ACCEDER A UN SERVIDOR

Otra utilidad interesante es la posibilidad de acceder a un servidor para enviar o recibir archivos. Presentamos un pequeño ejemplo de apertura de conexión con un servidor determinado. Simplemente, mostramos cómo acceder después de introducir el nombre de usuario y la contraseña.

En primer lugar, debemos pedir la dirección del servidor para poder abrir el archivo de comunicación:

```
Servidor$ = Input
("¿Dirección del Servidor? ")
Comunicacion=OpenTCPStream
(Servidor$,21)
```

El siguiente paso, después de detectar si ha habido conexión, es pedir el nombre de usuario. Éste se enviará precedido de la directiva "USER" necesaria para que el servidor reconozca el tipo de orden enviada:

```
If Not Comunicacion RuntimeError
("ERROR EN LA CONEXION")
Usuario$=Input$("¿Nombre de
usuario? ")
Orden$="USER"+Usuario$
WriteLine Comunicacion,Orden$
```

Hacemos la misma operación con la contraseña, esta vez con la directiva "PASS":

```
Pass$=Input$("¿Contraseña? ")
Orden$="PASS"+Pass$
WriteLine Comunicacion,Orden$
```

Una vez realizadas todas las operaciones de enlace con éxito pasamos al bucle principal, el cual utilizamos para enviar todas las órdenes que queramos al servidor:

```
Repeat
  Mensaje$=ReadLine(Comunicacion)
  Print Mensaje$
  Orden$=Input("¿Operacion? >")
  WriteLine Comunicacion,Orden$
Until Orden$="Fin"
```

Como podemos comprobar este tipo de operaciones es realmente sencillo y proporciona una gran flexibilidad para desarrollar aplicaciones para internet.

DIRECTPLAY

No queremos dejar el uso de las DirectPlay con las funciones para TCP sin mostrar un sencillo ejemplo de cómo crear un juego multijugador con ellas. La idea básica es mostrar en pantalla un jugador local y uno remoto representados por una pequeña esfera de distinto color. Ambos son movidos por el usuario con los cursores. El jugador local verá su esfera moverse en su pantalla y en la del otro ordenador y viceversa. Al principio creamos un jugador local y cuando el remoto se una creamos otro con diferente gráfico. Éste se moverá siguiendo las coordenadas recibidas a través de la red; es decir, cada jugador envía al otro equipo sus coordenadas X e Y, y ambos las utilizan para mover al jugador remoto. Para empezar, y después de definir las estructuras para cada jugador, debemos preguntar quién hará de anfitrión; es decir, quién creará el juego, lo cual almacenamos en la variable "Host". En el ejemplo no usaremos la ventana estándar de las DirectPlay de Windows. Para ello, utilizamos la función "HostNetGame"

```
PosBank=CreateBank(2)
........
Function ByteS$(valor)
 PokeByte PosBank,0,valor
 For n=0 To 1
  S$=S$+Chr(PeekByte(PosBank,n))
 Next
 Return S$
End Function
```

1

Función para extraer los números de un string y convertirlos a sus valores numéricos.

Código 3. Entramos en un bucle principal

```
JugLocal\x=JugLocal\x+(KeyDown(205)-KeyDown(203))
JugLocal\y=JugLocal\y+(KeyDown(208)-KeyDown(200))
DrawImage JugLocal\Ent,JugLocal\x,JugLocal\y
Text JugLocal\x,JugLocal\y-10,JugLocal\nombre$
Posicion$=ByteS$(JugLocal\x)+ByteS$(JugLocal\y)
SendNetMsg 1, Posicion$,JugLocal\ID,0
```

en vez de "StartNetGame" (Ver Código 2).

En caso de que el usuario ha elegido ser el anfitrión (Host) pasamos a crear el juego con el nombre "Nombre_juego$". En caso contrario, utilizamos "JoinNetGame" para unirnos al juego creado. Para ello, necesitamos también saber la dirección IP del equipo donde está el nuevo juego, la cual hemos preguntado con anterioridad y que almacenamos en "IPServ$". Ya tenemos preparado al sistema para albergar a cada jugador. Cada ordenador muestra a su jugador, por lo tanto, debemos crear un jugador local:

```
JugLocal.JugadorLocal =
New JugadorLocal
JugLocal\Ent = LoadImage
              ("Jug1.png")
```

```
JugLocal\x = 100
JugLocal\y = 100
JugLocal\ID = CreateNetPlayer
              (nombre$)
JugLocal\nombre$ = nombre$
```

De este jugador poseemos su entidad, sus coordenadas y su identificación en el juego. Para crear el jugador en el sistema multijugador es necesario utilizar "CreateNetPlayer". Una vez creado el jugador local entramos en un bucle principal donde movemos al jugador con los cursores y enviamos su posición al otro sistema con un mensaje de tipo 1 (Ver Código 3).

Observamos cómo el sistema de DirectPlay recibe y envía mensajes de carácter alfanumérico, por lo tanto, debemos convertir las coordenadas numéricas en string. Para ello, utilizamos la función "ByteS$" cuya finalidad es convertir cada valor en alfanumérico utilizando como intermediario un banco de memoria. (Fig. 1).

Continuamos con las instrucciones necesarias para recibir los mensajes del otro ordenador y obrar en consecuencia. Esta operación la realizamos con la función "RecNetMsg()". Los mensajes utilizados por las funciones de DirectPlay vienen automáticamente encabezados por un código que indica de qué tipo es (Ver Fig. 2).

Así que, utilizando una estructura "Select..Case" realizamos las operaciones según el tipo de mensaje (Ver Código 4).

Si recibimos un código 100 quiere decir que un nuevo jugador

```
Tipo 1 .. 99
  MENSAJE DE USUARIO
Tipo 100
  UN JUGADOR SE HA UNIDO
Tipo 102
  EL HOST ORIGINAL DEJA LA PARTIDA
Tipo 200
  HA OCURRIDO UN ERROR GRAVE
```

2

Descripción de los diferentes tipos de mensajes producidos por NetMsgType.

Código 2. Nos preguntaremos quién hará de host

```
If host
  Text 220,260,"CREANDO JUEGO "+Nombre_Juego$
  n=CountHostIPs("")
  IP = HostIP(n)
  If HostNetGame (Nombre_Juego$)=2
      Text 220,280,"JUEGO "+Nombre_Juego+" CREADO CORRECTAMENTE"
      Delay 1000
    Else
      RuntimeError "No se ha podido iniciar un nuevo juego"
  EndIf
Else
  If JoinNetGame (Nombre_Juego$,IPServ$)=0
    RuntimeError "No se ha podido unir al juego"
  Else
    AppTitle "UNIDO AL JUEGO "+Nombre_Juego+" CORRECTAMENTE"
  EndIf
EndIf
```

```
PosBank=CreateBank(2)
.........
Function StringS(St$)
 For n=0 To 1
  PokeByte PosBank,n,Asc(Mid(St$,n+1,1))
 Next
 Num=PeekShort (PosBank,0)
 Return Num
End Function
```

3

Función para convertir un número en un string alfanumérico.

se ha unido a la partida, por lo que tenemos que crearlo. A partir de aquí si no ocurre ningún error (tipo 200) recibiremos mensajes de tipo 1 conteniendo las coordenadas del jugador remoto. Así que solo tenemos que convertir este mensaje en números para obtener las coordenadas y actualizarlo.

La función "StringS" simplemente realiza la operación contraria a la operación "ByteS$"; es decir, convierte caracteres en números (Ver Fig. 3).

AMPLIACIÓN DEL LENGUAJE

Blitz3D nos proporciona la opción de ampliar sus posibilidades a través del uso de librerías de enlace dinámico "DLL" programadas con otros lenguajes como C, C++, Visual Basic, PureBasic, etc. Para llamar a las funciones contenidas en una DLL, disponemos de la instrucción "CallDLL":

CallDLL (nombre_libreríaDLL, Nombre del procedimiento, banco de entrada, banco de salida).

Los bancos de memoria son utilizados para el paso de parámetros entre la librería y Blitz3D. El Código 5 muestra un sencillo ejemplo de creación de una DLL en Visual C para visualizar un texto en una ventana de Windows y su posterior utilización por Blitz3D.

Como podemos comprobar, para construir una librería de enlace dinámico se necesitan conocimientos más avanzados de programación en otros lenguajes. El ejemplo que hemos presentado solo pretende servir como referencia para aquellos con el interés de realizar sus propias librerías DLL.

Blitz3D llegará muy pronto con una nueva ampliación denominada BlitzMax. Este nuevo lenguaje basado en Blitz3D admite nuevas características en las que se incluye la posibilidad multiplataforma y la habilidad de compilar código escrito en OpenGl directamente desde el editor.

Esperamos que Blitz3D cubra todas vuestras necesidades de desarrollo y que pronto podamos ver entre los seguidores de este curso un amplio abanico de nuevos juegos con una calidad admirable.

Código 4. Realizamos las operaciones según el tipo de mensaje

```
If RecvNetMsg()
  Select NetMsgType()
    Case 100:
      From=NetMsgFrom()
      JugRemoto.JugadorRemoto = New JugadorRemoto
      JugRemoto\Ent=LoadImage("Jug2.png")
      JugRemoto\x=200
      JugRemoto\y=200
      JugRemoto\ID=From
      JugRemoto\nombre$=NetPlayerName(From)
    Case 101:
      AppTitle "EL JUGADOR REMOTO SE FUE"
      Delete JugRemoto
    Case 200:
      RuntimeError "UN ERROR GRAVE HA OCURRIDO"
    Case 1:
      Coordenadas$=NetMsgData()
      RemX=StringS(Left (Coordenadas$,2))
      RemY=StringS(Right(Coordenadas$,2))
      For JugRemoto.JugadorRemoto=Each JugadorRemoto
        JugRemoto\x=RemX
        JugRemoto\y=RemY
        DrawImage JugRemoto\Ent,JugRemoto\x,JugRemoto\y
        Text JugRemoto\x,JugRemoto\y,JugRemoto\nombre$
      Next
  End Select
EndIf
```

Código 5. Creación de una DLL en Visual C

■ Cabecera del procedimiento:

```
#include "windows.h"
extern "C" {
  _declspec (dllexport) int _cdecl ejemplo (int const void *int,
  int TamañoEntrada, int *out, int TamañoSalida);
          }
```

■ Procedimiento:

```
Int ejemplo (int const *in, TamañoEntrada, int *out TamañoSalida)
{
  MessageBox (FindWindow (NULL, "ProcDLL"),"Esto es un ejemplo
  del uso de DLL en Blitz3D", "DLL en Blitz3D", MB_Ok);
  return (1);
}
```

■ Desde Blitz3D:

```
AppTitle "Proc. DLL"
Llamada=CallDLL ("Procedimiento","ejemplo")
WaitKey()
```

Instalación del juego con InnoSetup

Cuando se termina de desarrollar un videojuego normalmente se tienen multitud de carpetas y docenas de archivos situados en un lugar determinado del disco duro que el usuario final desconoce.

Por lo tanto, resulta primordial incluir un sistema de instalación guiada en el que el jugador solo tenga que responder a una serie de preguntas para adaptar toda esa información en su equipo. Otra finalidad de un sistema de instalación es empaquetar todas las carpetas y archivos del juego en un solo fichero (generalmente comprimido) para facilitar su distribución.

En el mercado existe multitud de instaladores. Los encontramos de pago y algunos de libre uso. Para "Zone of Fighters" hemos elegido uno de los mejores instaladores gratis que podemos encontrar hoy día, Inno Setup.



CARACTERÍSTICAS

Muchas son las cualidades de este instalador, pero lo que más destaca es su estabilidad y la cantidad de posibilidades que admite. Entre ellas, su sencillez de uso. Puede funcionar en todas las versiones de Windows incluso para NT 4.0 o XP. Inno Setup genera un script con todos los procesos de la instalación para que podamos programarlo a nuestro gusto. Además, admite compresión de archivos, creación de atajos en cualquier lugar del sistema (barra de tareas, escritorio) y hasta creación de registros de sistema.

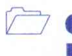
En nuestro tutorial solo realizaremos una instalación básica ayudados por el sistema de creación guiado del programa o "Wizard". Aun así, es suficiente para nuestros propósitos.



DEFINICIÓN

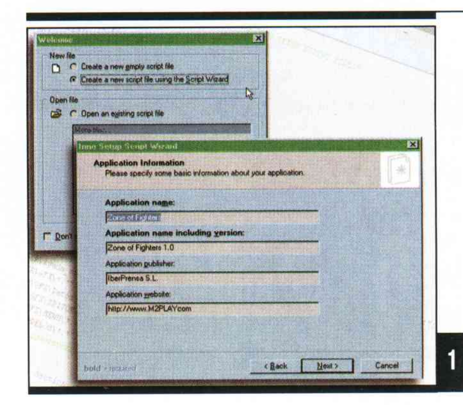
▸ SCRIPT

Un *Script* es un fichero de texto con un formato determinado parecido a un lenguaje de programación que puede ser editado. Es propio de la aplicación que lo utiliza y sirve para programar su funcionamiento.



CREANDO LA INSTALACIÓN

Al ejecutar Inno Setup, el programa nos pregunta si queremos escribir un Script directamente o si preferimos utilizar la instalación guiada. De cualquier forma, al final obtendremos en el editor del programa un texto de script con la instalación, el cual podemos modificar posteriormente a nuestro gusto. Seleccionamos la opción *Create a new script file using the ...* y en la siguiente ventana pulsamos en *Next*. El siguiente diálogo se refiere a la información del programa. En la primera escribimos el nombre del juego y en la segunda la versión, por ejemplo "Zone of Fighters 1.0". La tercera y cuarta casilla son para el nombre y la web de quien publica el juego (Fig. 1).

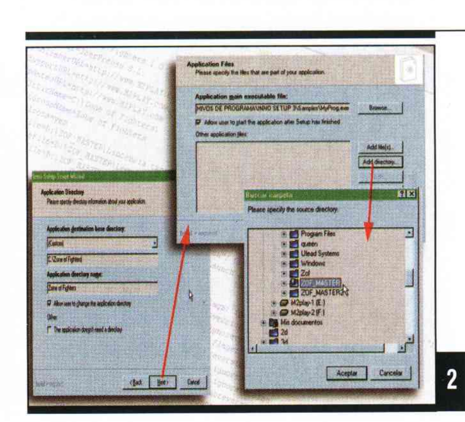


El primer paso es asignar los nombres de la instalación.

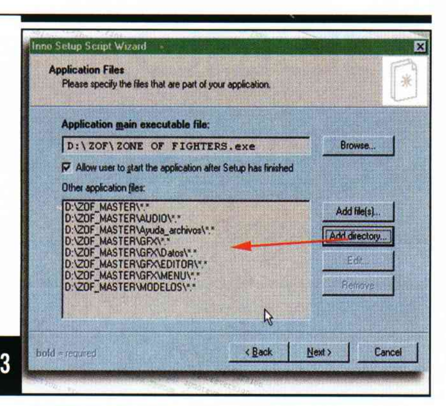
Luego entramos en el diálogo sobre el directorio donde se instalará el juego. Por defecto, el programa nos sitúa en *Archivos de programa* (*Program Files*). En nuestro caso, el juego utiliza una ubicación fija y establecida de antemano en el código, *C:\Zone of Fighters*. Por lo tanto, debemos seleccionar esta ubicación para que el instalador coloque ahí todos los ficheros. Puede ocurrir que haya usuarios que no tengan nombrado su disco duro principal como "C:". En tal caso no podrían instalar el juego. Así que debemos permitirle que elija la suya, ya que mantener una ubicación fija no resulta nada profesional. Para modificar este aspecto en "Zone of Fighters" tendremos que cambiar el código de esta forma:

Por ejemplo, sustituimos:

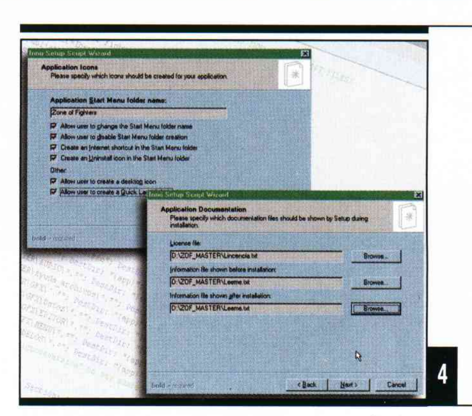
```
bionave=LoadMesh ("c:\zone of
fighters\modelos\bionave.3ds")
```



Podemos asignar archivos individuales o carpetas completas.



Es importante seguir el orden en la suma de subcarpetas.



Podemos definir accesos directos tanto en el escritorio como en la barra de tareas.

o

```
control=LoadImage("c:\zone of
fighters\gfx\menu\control.png")
```

por

```
bionave=LoadMesh (" \modelos\
bionave.3ds")
```

y

```
control=LoadImage("\gfx\menu\
control.png")
```

etc.

Si elegimos esta última opción tendremos que dejar seleccionada la casilla *Allow user to change ....* para que el usuario pueda cambiar la ubicación del juego. El siguiente paso es decirle al instalador cuál será el ejecutable del juego (el cual elegimos a través del explorador pulsando en el botón "Browse...") y dónde está. Si nuestro juego posee además otros ficheros o carpetas tendremos que incluirlos a través de *Other Application files*. Éste es el trabajo más duro, ya que tenemos que ir sumando todas las carpetas y subcarpetas.

Supongamos que tenemos todo el juego en la carpeta "ZOF_MASTER". Empezamos añadiendo esta capeta. Pulsamos en *Add directory...* y la elegimos desde el explorador. Hacemos lo mismo con todas las demás. Si tenemos una carpeta con subcarpetas, tendremos que elegir por orden la carpeta y seguidamente cada una de las demás subcarpetas (Fig. 2 y 3).

Una vez añadidos todos los ficheros del juego pasamos a definir la generación de iconos. Automáticamente, el programa coloca el nombre del juego dado anteriormente para la barra de tareas de Windows (casilla *Application Start Menu ...*). Si queremos que el instalador cree un acceso directo del juego y de la página web relacionada con él previamente definida, además de que el usuario pueda decidir si quiere o no un acceso directo, seleccionaremos todas las casillas de esta ventana. Por último, el siguiente paso es incluir los ficheros de documentación del juego. La primera casilla se refiere al fichero de texto que explica el tipo de licencia del juego. La siguiente mostrará un fichero de texto antes de empezar la instalación y la última casilla es para un fichero de texto después de la instalación (Fig. 4).

A la pregunta de si queremos ejecutar la compilación del nuevo script respondemos que no.

 EL COMPILADOR

Después de crear nuestra instalación aparecerá en el editor del compilador el script generado por el sistema guiado, el cual podemos modificar a nuestro gusto. Observamos que el código está dividido en las diferentes partes que hemos estado rellenando durante la creación con el Wizard.

En "Setup" disponemos de todos los datos referentes a la instalación como el nombre del juego, detalles del editor, de la ubicación del programa y de los textos de información. En "Task" se muestran las características activadas para la creación. En "Files" se detalla la situación de los archivos para crear el ejecutable de instalación definitivo (*Source*) y dónde serán ubicados durante la ejecución de la instalación (*DestDir*). La sección "INI" se refiere a los ficheros de entrada instalados en el sistema. En "Icons" se define la ubicación y nombre de los iconos creados y en "Run" la ubicación del programa que ejecutará después de terminar la instalación.

Para crear el fichero de instalación debemos pulsar en el icono o bien con la opción *Run* (F9). Si todo sale bien se habrá creado un único fichero ejecutable en el directorio *Output* creado en la misma carpeta donde tenemos los ficheros del juego.

.. Esperamos que todos los tutoriales expuestos en el curso hayan ampliado aún más vuestros conocimientos para llevar a cabo mejores desarrollos.



NOTA

Cuando ejecutemos una instalación realizada con Inno Setup, todas las carpetas y archivos se ubicarán de la misma manera que fueron incluidas al crear la instalación.



NOTA

La expresión {app} en el script significa el directorio del juego el cual fue elegido en la casilla *Application destination base directory*.

Simuladores de velocidad (y II)

Los simuladores de velocidad no se han limitado al mundo de las cuatro ruedas. Paralelo a éstos se han desarrollado títulos para el mundo del motociclismo y otras disciplinas.

Además, en la mayoría de los casos, se han sustituido los vehículos clásicos por otros más futuristas con características de ensueño. Incluso algunos han mezclado la velocidad con el combate. En esta entrega hablaremos un poco de todos ellos.

✍ SOBRE DOS RUEDAS

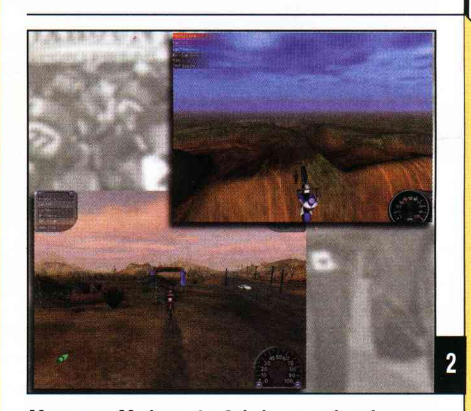
La disciplina del motor siempre ha estado vinculada también a los vehículos de dos ruedas, bien en competiciones en circuitos cerrados como en espacios abiertos (motocross) como ocurre con el automovilismo. No se han desarrollado tantos títulos, y menos de calidad, para esta disciplina como ocurre con la fórmula 1 o el rally, pero de igual manera empezó su producción para las antiguas consolas de videojuegos y ordenadores de 8 bits que intentaban recrear a las recreativas de la época. La premisa técnica básica era exactamente la misma que la utilizada para simular el automovilismo, la única diferencia radicaba en la representación de los vehículos. A medida que avanzaban las capacidades técnicas y el uso de dispositivos de juegos se diferenciaron las características del comportamiento físico del vehículo a la respuesta del jugador. Los juegos de motociclismo se han orientado siempre al concepto arcade en torno a las competiciones con más renombre mundial, ya que las posibilidades de personalización de los vehículos son casi nulas comparadas con las de las 4 ruedas.

La primera referencia para juegos de motos en el PC la encontramos en el título *Motocross* de Microsoft en 1989 para MS-DOS. Al igual que ocurría con *Pole Position* en 1983, era en 2D y la acción avanzaba hacia la pantalla con el jugador situado en el centro y de espaldas. Un año después aparece *African Trail Simulator* de Positive con un poco de lo mismo. En la modalidad de racing se desarrollaron más títulos destacando en 1989 *The Cycles: International Grand Prix Racing* (Distinctive Software/ Accolade) con un jugabilidad arcade estupenda con posibilidad de elegir diferentes cilindradas. Este mimo año aparece un nuevo concepto de motociclismo en la simulación de la legendaria *Harley Davidson* en la que la ciudad adquiere matiz de circuito con *Harley DavidSon: The Road to Sturges* (Incredible Technologies, Inc). A este título le siguieron otros de la misma serie hasta el 2000. Tuvieron que pasar algunos años para encontrar un juego que realmente llenara el mundo del motociclismo para PC. La desarrolladora Delphine Software International, aprovechando las características de la aceleración gráfica 3Dfx, las MMX de Intel junto a las posibilidades de la comunicación vía MODEM, realiza *Moto Racer* (Electronics Arts, 1996). Poseía dos estilos de competición superbike y motocross en 8 circuitos perfectamente diseñados. Soportaba hasta 8 jugadores conectados y fue el primero en incluir sonido 3D. *Moto Racer* es ya un clásico y se han realizado tres partes más hasta el 2002 con *Moto Racer 3*. A este título le siguieron otros que simulaban



1

(Arriba) The Cycles: International Grand Prix Racing. (Abajo) Moto Racer.


2

Motocross Madness 1 y 2: la locura sobre dos ruedas.



3

El realismo conseguido con la serie Superbike es increíble.



4

Los mejores exponentes de la competición de 500 cc. (Arriba) GP 500. (Abajo) GP.



5

Clásicos en los juegos de velocidad futurista. (Arriba) MegaRacer 1, 2 y 3. (Abajo) Wipeout 1 y Wipeout Fusion.



6

Imagen de Doom III de ID Software, un ejemplo del futuro próximo que nos espera para juegos de ordenador.

sus cualidades. El más destacado fue *Motocross Madness* (1998) de Microsoft con uso de las Direct3D; poseía gráficos en 3D real con soporte para dispositivos de juegos especializados y todas las conexiones diferentes para multijugador. La segunda parte, publicada en el 2000, es todo un espectáculo visual en el que puedes elegir varios tipos de competición, preparar la moto y jugar con otros jugadores por red.

En este mismo año se versiona para PC *Jeremy McGrath Supercross 2000,* el cual ofrecía la novedad de incluir un editor de circuitos. A partir del 2000 los títulos alcanzan un realismo asombroso y pocas son las cualidades de que carecen. El aprovechamiento de las 3D y las posibilidades gráficas del momento irrumpieron en todas las publicaciones. Antes del lanzamiento de F1 2000 con la licencia FIA de automovilismo, EA Sports publica *SuperBike World ChampionShip* a finales de 1998 con la licencia del mundial SBK, compitiendo con el *Motocross Madness* de Microsoft. Las cualidades técnicas eran casi idénticas ya que poseía gráficos en 3D real o animación por Motion Capture. *SuperBike* contempla unas cualidades casi perfectas para un juego de estas características que proporcionan un realismo impresionante. Se trataba pues de un digno simulador en el que incluso se puede observar el movimiento de los pilotos en las curvas y el cambio de sonido en cada marca. La siguiente entrega, *SuperBike 2000,* aunque con mayores características técnicas como más vistas de cámara, movimientos o más detalles en primera persona, no superó en absoluto a su versión anterior. Por otro lado, queremos terminar con la disciplina rey en circuitos de velocidad, los 500 cc. Este evento encontró una buena salida en el mercado con títulos como la serie *Grand Prix 500* (MicroProse / Hasbro 1999). Pero sin duda, el mejor acercamiento a esta disciplina estuvo a cargo de Namco cuando versiona su recreativa de 1999 *Namco 500cc GP* para consola con el nombre de *Moto GP* y que posteriormente también pasa al PC. *Moto GP* está considerado el mejor simulador de 500cc. de hoy en día. Su realismo es total y alcanza hasta en la maniobrabilidad de las motos. Es el primero en añadir el freno en las dos ruedas e inclinación de la moto a voluntad. Proporciona también cambios climatológicos que afectan al manejo de la moto y sistema de Motion Capture en los pilotos.

VELOCIDAD FUTURISTA

El concepto de las carreras en los videojuegos también se extiende en el campo de la ciencia-ficción ofreciendo títulos con un talante de arcade de lo más puro. Uno de los primeros intentos por simular carreras ambientadas en el espacio o con vehículos futuristas para PC corre a cargo de Electronics Arts con la versión de *Powerdrome* de Amiga en el que se simulaba una carrera de naves por un circuito cerrado. Inspirado en la realidad virtual aparece en 1993 *MegaRace* (Cryo Interactive Entertainment) con gráficos en 3D, vídeo Full Motion y voz. Siguió una secuela con *MegaRace 2* (1996) y *MegaRace 3* (2002). Pero el título que ha encabezado las listas de simuladores de carreras futuristas ha sido la serie *Wipeout* de Psygnosis desde 1995. Fue el primer título de Psygnosis para Nintendo 64 y que versionó posteriormente para PC. Con *Wipeout* se rompen moldes en cuestión de jugabilidad ofreciendo una velocidad de juego endiablada y el primero con características multijugador. Además de estos títulos, no podemos olvidarnos de otros de gran calidad como la serie *Start Wars: Episodio I: Racer* (Lucas Arts Entertainment, 1999), basado en la película del mismo nombre.

... Cada día la tecnología aplicada a los juegos avanza más y más. ¿Qué maravillas nos deparará el futuro? La pregunta queda en el aire...

Cuestionario Videojuegos 20

Preguntas

1. ¿Cómo podemos obtener el nombre de un equipo conectado a la red local?

2. ¿Con qué instrucción abrimos un archivo de escritura en un servidor?

3. ¿Qué es lo primero que debemos realizar antes de clasificar los mensajes recibidos por el protocolo UDP?

4. ¿Cómo podemos enviar una variable de 4 bytes a través del protocolo UDP?

5. En Deep Paint 3D, ¿cómo podemos obtener una ventana con la textura?

6. ¿Con qué herramienta podemos dibujar una silueta curva en Deep Paint 3D?

7. ¿Qué teclas del teclado equivalen a las notas DO y RE# en ModPlug?

8. ¿Cómo podemos activar el uso de acordes en ModPlug?

9. ¿Cómo podemos permitir en Inno Setup que el usuario pueda cambiar el destino de la instalación?

10. En Inno Setup, ¿en qué parte del Script se encuentra lo referente a los ficheros contenidos en la instalación?

Respuestas al cuestionario 19

▷ **1.** Num_IP = HostIP(CountHostIps(""))
Direccion_IP$ = DottedIP$ (Num_IP)

▷ **2.** Crear juego:
Nuevo_Juego = HostNetGame (" PARTIDA 1")
Unirse a un juego:
IP_Host = "127.0.0.1"
Nueva_Union = JoinNetGame ("PARTIDA 1", IP_Host)

▷ **3.** Type botones
Field grafico
Field x,y
End Type
Dim boton.botones(9)
Dim botonB.botones(9)

▷ **4.** Utilizando la instrucción "ImagesOverlap":
If ImagesOverlap(cursor,MouseX(),MouseY(),botonB\grafico,botonB\x,botonB\y)
DrawImage boton\grafico,botonB\x,botonB\y
If MouseDown(1) eleccion=n Else eleccion=-1
EndIf

▷ **5.** Elegimos en *Styles* los gradientes. Dentro de la ventana de diálogo elegimos el tipo esférico y entramos en *Edit*. En la ventana de *Edit* pulsamos el botón "New" y en Gradient seleccionamos los dos puntos de color.

▷ **6.** Que el gradiente nunca llegue a cubrir toda la imagen.

▷ **7.** En la sección General pulsamos el botón *Change* en la sección *Song Type* y elegimos los canales.

▷ **8.** En la sección *Samples* pulsamos en el icono "New" y seguidamente cargamos la muestra. Cada vez que se pulsa en "New" se crea una plantilla vacía a la espera de recibir un archivo de muestra.

▷ **9.** Haciendo clic sobre la línea horizontal que atraviesa el clip de audio en el punto dónde queremos modificar el volumen y desplazando el punto creado.

Respuestas al cuestionario 20

▷ **1.** Utilizando la instrucción "GetEnv" ➠ GetEnv("LocalHost")

▷ **2.** Con la instrucción "OpenTCPStream" ➠ Comunicación=OpenTCPStream(Dirección$, Puerto)

▷ **3.** Leer el mensaje con "RecvUDPMsg" ➠ Mensaje=RecvUDPMsg(UDP_Stream)

▷ **4.** Utilizando "WriteFLoat" ➠ WriteFloat UDP_Stream, valor#

▷ **5.** Haciendo doble clic sobre el canal "C" en la capa.

▷ **6.** Utilizando *Splines*. Una vez trazada la curva se pulsa con el botón derecho sobre ella y se elige la opción *BrushStroke Curve* con un pincel seleccionado.

▷ **7.** El DO corresponde a la tecla Q y el RE# a la R.

▷ **8.** Si el acorde es de dos notas haciendo clic con la tecla "SHIFT" pulsada sobre dos canales.

▷ **9.** Activando la casilla *Allow user to change the aplication directory*.

▷ **10.** La relación de ficheros se encuentra después del encabezamiento [Files].

Contenido CD-ROM 20

AUDIO

AudioMaestro
Serie de utilidades para editar música, crear cds y un montón de cosas más.

Music Collector
Ten bien ordenada tus músicas y colecciónalas usando esta aplicación.

Micoco MP3 to Wav converter Plus
Convierte fácil y rápidamente tus MP3 en archivos WAV.

RecordNow Max
Excelente software de grabación de cds con un montón de opciones.

Sound Probe



Con este estupendo programa podrás editar tus archivos de audio y añadirles efectos.

CD MP3 Terminator
Graba tus MP3 a cd de un modo muy sencillo usando esta programa.

DISEÑO 2D

XPhoto Lite



Excelente programa con el que podrás añadir múltiples efectos a tus imágenes.

Irfanview
Gracias a esta aplicación podrás previsualizar tus imágenes rápidamente.

Poweralbums
Ten siempre tus fotografías y dibujos ordenador con este catalogador.

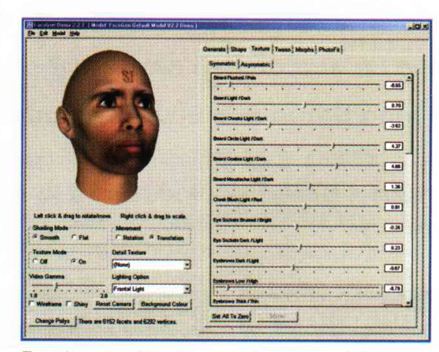
Slowview
Otro software de catalogación con el que además podrás visualizar tus imágenes.

VCW VicMan's Photo Editor
Programa de edición de imágenes para realizar los cambios que desees.

DISEÑO 3D

Ayam
Programa de edición y desarrollo de imágenes en tres dimensiones.

FaceGen



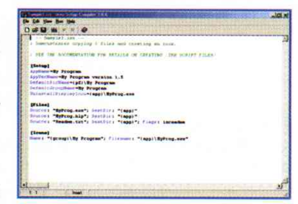
Excelente aplicación con la que podrás caracterizar toda clase de rostros a tu gusto.

Nendo
Demo de esta herramienta con la que podrás crear entornos y objetos en 3D.

Rhinoceros
Interesante utilidad para desarrollar escenas en tres dimensiones viéndolas además a la vez en varias perspectivas.

PROGRAMACIÓN

Inno Setup
Nueva oportunidad para conseguir este excelente software con el que realizaremos el instalador del juego.



NCTDiscWriter ActiveX DLL
Gracias a esta herramienta podrás crear backups de tus datos y no perder nada.

Emeditor
Editor muy cómodo de usar que soporta además múltiples lenguajes de programación.

JUEGOS

Motocross Madness 2
Haz que tus competidores muerdan el polvo de la derrota con este divertido juego.

Motocross Mania
Trepidante simulador en el que no tendrás descanso alguno hasta llegar a la meta.

Tux Racer
Simpático juego completo en el que deberás correr una trepidante carrera deslizándote por la nieve.

AmaSuperBike
Corre con tu moto y diviértete de lo lindo en todas las carreras que desees.

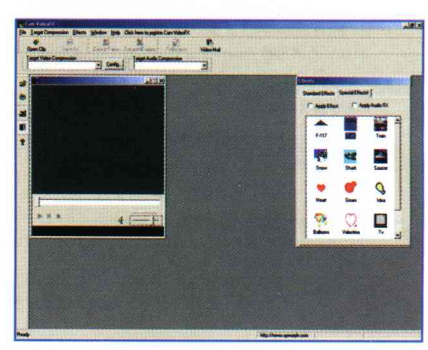


Star Wars Episode 1 Racer
Competición de tipo futurista en la que además podrás meterte en la piel de los personajes de Star Wars.

Zone of Fighters
Nuestro juego, en su versión ya definitiva. Además incluimos todo el código fuente.

VÍDEO

CamVideo FX



Interesante utilidad con la que podrás pasar tus vídeos desde la cámara digital y un montón de utilidades más.

X Audio Video Clip Joiner
Combina fácil y rápidamente los clips de vídeo con el audio.

▶ EXTRAS
En este apartado encontrarás todos los ejemplos de los que hablamos en el coleccionable, para que no pierdas detalle.